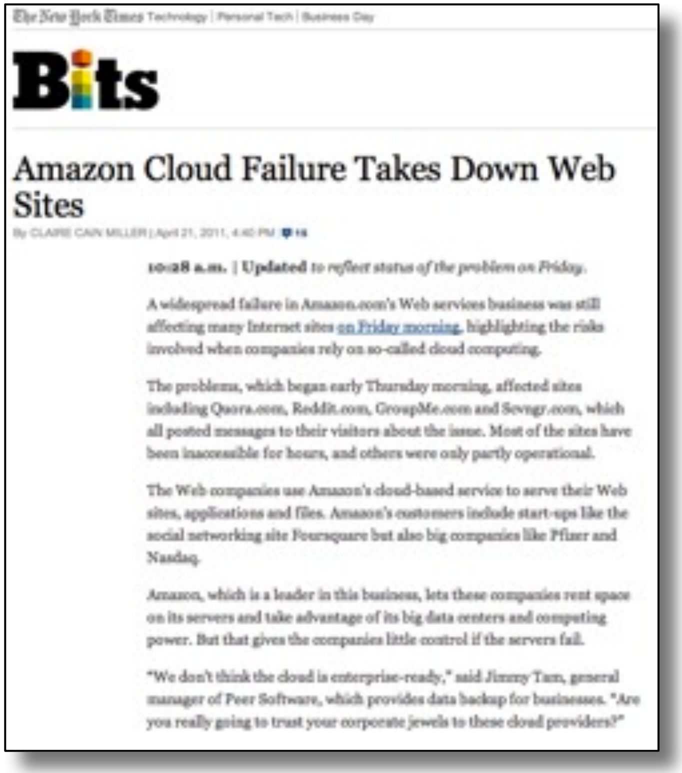
Consistent Updates for Software-Defined Networks: Change You Can Believe In!

Mark Reitblatt, Nate Foster,
Jen Rexford, and Dave Walker



The New York Times Technology | Personal Tech | Business Day

Bits

Amazon Cloud Failure Takes Down Web Sites

By CLAIRE CAIN MILLER | April 21, 2011, 4:40 PM 18

10:28 a.m. | Updated *to reflect status of the problem on Friday.*

A widespread failure in Amazon.com's Web services business was still affecting many Internet sites on Friday morning, highlighting the risks involved when companies rely on so-called cloud computing.

The problems, which began early Thursday morning, affected sites including Quora.com, Reddit.com, GroupMe.com and Scvngr.com, which all posted messages to their visitors about the issue. Most of the sites have been inaccessible for hours, and others were only partly operational.

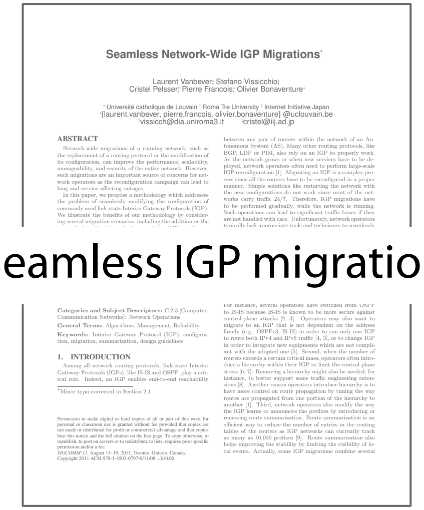
The Web companies use Amazon's cloud-based service to serve their Web sites, applications and files. Amazon's customers include start-ups like the social networking site Foursquare but also big companies like Pfizer and Nasdaq.

Amazon, which is a leader in this business, lets these companies rent space on its servers and take advantage of its big data centers and computing power. But that gives the companies little control if the servers fail.

"We don't think the cloud is enterprise-ready," said Jimmy Tam, general manager of Peer Software, which provides data backup for businesses. "Are you really going to trust your corporate jewels to these cloud providers?"

"[A] network change was performed as part of our **normal** AWS scaling activities...
This change **disconnected** both the primary and secondary network simultaneously, leaving the affected nodes completely isolated from one another."

Prior Work



Seamless Network-Wide IGP Migrations*

Laurent Vanbever*, Stefano Vissicchio†,
Cristel Pelsser‡, Pierre Francois*, Olivier Bonaventure*

* Université catholique de Louvain † Roma Tre University ‡ Internet Initiative Japan
*{laurent.vanbever, pierre.francois, olivier.bonaventure} @uclouvain.be
†vissicch@dia.uniroma3.it ‡cristel@iij.ad.jp

ABSTRACT

Network-wide migrations of a running network, such as the replacement of a routing protocol or the modification of its configuration, can improve the performance, scalability, manageability, and security of the entire network. However, such migrations are an important source of concerns for network operators as the reconfiguration campaign can lead to long and service-affecting outages.

In this paper, we propose a methodology which addresses the problem of seamlessly modifying the configuration of commonly used link-state Interior Gateway Protocols (IGP). We illustrate the benefits of our methodology by considering several migration scenarios, including the addition or the

Categories and Subject Descriptors: C.2.3 [Computer-Communication Networks]: Network Operations

General Terms: Algorithms, Management, Reliability

Keywords: Interior Gateway Protocol (IGP), configuration, migration, summarization, design guidelines

1. INTRODUCTION

Among all network routing protocols, link-state Interior Gateway Protocols (IGPs), like IS-IS and OSPF, play a critical role. Indeed, an IGP enables end-to-end reachability

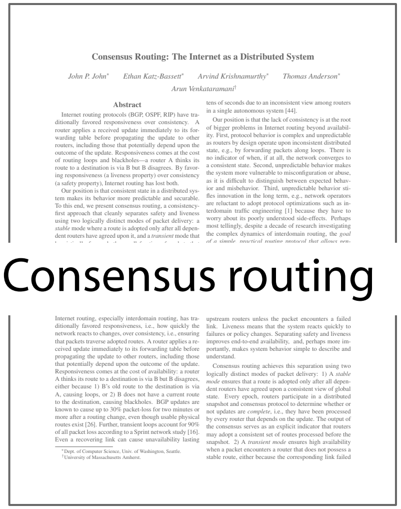
between any pair of routers within the network of an Autonomous System (AS). Many other routing protocols, like BGP, LDP or PIM, also rely on an IGP to properly work. As the network grows or when new services have to be deployed, network operators often need to perform large-scale IGP reconfiguration [1]. Migrating an IGP is a complex process since all the routers have to be reconfigured in a proper manner. Simple solutions like restarting the network with the new configurations do not work since most of the networks carry traffic 24/7. Therefore, IGP migrations have to be performed gradually, while the network is running. Such operations can lead to significant traffic losses if they are not handled with care. Unfortunately, network operators typically lack appropriate tools and techniques to seamlessly

to IS-IS because IS-IS is known to be more secure against control-plane attacks [2, 3]. Operators may also want to migrate to an IGP that is not dependent on the address family (e.g., OSPFv3, IS-IS) in order to run only one IGP to route both IPv4 and IPv6 traffic [4, 3], or to change IGP in order to integrate new equipments which are not compliant with the adopted one [5]. Second, when the number of routers exceeds a certain critical mass, operators often introduce a hierarchy within their IGP to limit the control-plane stress [6, 7]. Removing a hierarchy might also be needed, for instance, to better support some traffic engineering extensions [8]. Another reason operators introduce hierarchy is to have more control on route propagation by tuning the way routes are propagated from one portion of the hierarchy to another [1]. Third, network operators also modify the way the IGP learns or announces the prefixes by introducing or removing route summarization. Route summarization is an efficient way to reduce the number of entries in the routing tables of the routers as IGP networks can currently track as many as 10,000 prefixes [9]. Route summarization also helps improving the stability by limiting the visibility of local events. Actually, some IGP migrations combine several

[*] Minor typo corrected in Section 2.1

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGCOMM'11, August 15–19, 2011, Toronto, Ontario, Canada.
Copyright 2011 ACM 978-1-4503-0797-0/11/08 ...$10.00.

Seamless IGP migration



Consensus Routing: The Internet as a Distributed System

*John P. John** *Ethan Katz-Bassett** *Arvind Krishnamurthy** *Thomas Anderson**
Arun Venkataramani†

Abstract

Internet routing protocols (BGP, OSPF, RIP) have traditionally favored responsiveness over consistency. A router applies a received update immediately to its forwarding table before propagating the update to other routers, including those that potentially depend upon the outcome of the update. Responsiveness comes at the cost of routing loops and blackholes—a router A thinks its route to a destination is via B but B disagrees. By favoring responsiveness (a liveness property) over consistency (a safety property), Internet routing has lost both.

Our position is that consistent state in a distributed system makes its behavior more predictable and securable. To this end, we present consensus routing, a consistency-first approach that cleanly separates safety and liveness using two logically distinct modes of packet delivery: a *stable* mode where a route is adopted only after all dependent routers have agreed upon it, and a *transient* mode that

tens of seconds due to an inconsistent view among routers in a single autonomous system [44].

Our position is that the lack of consistency is at the root of bigger problems in Internet routing beyond availability. First, protocol behavior is complex and unpredictable as routers by design operate upon inconsistent distributed state, e.g., by forwarding packets along loops. There is no indicator of when, if at all, the network converges to a consistent state. Second, unpredictable behavior makes the system more vulnerable to misconfiguration or abuse, as it is difficult to distinguish between expected behavior and misbehavior. Third, unpredictable behavior stifles innovation in the long term, e.g., network operators are reluctant to adopt protocol optimizations such as interdomain traffic engineering [1] because they have to worry about its poorly understood side-effects. Perhaps most tellingly, despite a decade of research investigating the complex dynamics of interdomain routing, the *goal of a simple, practical routing protocol that allows gen-*

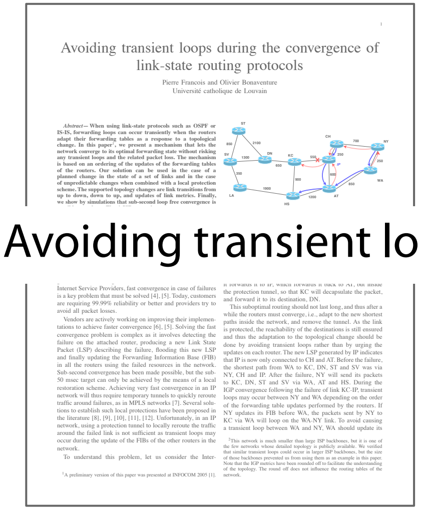
Internet routing, especially interdomain routing, has traditionally favored responsiveness, i.e., how quickly the network reacts to changes, over consistency, i.e., ensuring that packets traverse adopted routes. A router applies a received update immediately to its forwarding table before propagating the update to other routers, including those that potentially depend upon the outcome of the update. Responsiveness comes at the cost of availability: a router A thinks its route to a destination is via B but B disagrees, either because 1) B's old route to the destination is via A, causing loops, or 2) B does not have a current route to the destination, causing blackholes. BGP updates are known to cause up to 30% packet-loss for two minutes or more after a routing change, even though usable physical routes exist [26]. Further, transient loops account for 90% of all packet loss according to a Sprint network study [16]. Even a recovering link can cause unavailability lasting

upstream routers unless the packet encounters a failed link. Liveness means that the system reacts quickly to failures or policy changes. Separating safety and liveness improves end-to-end availability, and, perhaps more importantly, makes system behavior simple to describe and understand.

Consensus routing achieves this separation using two logically distinct modes of packet delivery: 1) A *stable mode* ensures that a route is adopted only after all dependent routers have agreed upon a consistent view of global state. Every epoch, routers participate in a distributed snapshot and consensus protocol to determine whether or not updates are *complete*, i.e., they have been processed by every router that depends on the update. The output of the consensus serves as an explicit indicator that routers may adopt a consistent set of routes processed before the snapshot. 2) A *transient mode* ensures high availability when a packet encounters a router that does not possess a stable route, either because the corresponding link failed

[*] Dept. of Computer Science, Univ. of Washington, Seattle.
[†] University of Massachusetts Amherst.

Consensus routing



1

Avoiding transient loops during the convergence of link-state routing protocols

Pierre Francois and Olivier Bonaventure
Université catholique de Louvain

Abstract—**When using link-state protocols such as OSPF or IS-IS, forwarding loops can occur transiently when the routers adapt their forwarding tables as a response to a topological change. In this paper[1], we present a mechanism that lets the network converge to its optimal forwarding state without risking any transient loops and the related packet loss. The mechanism is based on an ordering of the updates of the forwarding tables of the routers. Our solution can be used in the case of a planned change in the state of a set of links and in the case of unpredictable changes when combined with a local protection scheme. The supported topology changes are link transitions from up to down, down to up, and updates of link metrics. Finally, we show by simulations that sub-second loop free convergence is**

Internet Service Providers, fast convergence in case of failures is a key problem that must be solved [4], [5]. Today, customers are requiring 99.99% reliability or better and providers try to avoid all packet losses.

Vendors are actively working on improving their implementations to achieve faster convergence [6], [5]. Solving the fast convergence problem is complex as it involves detecting the failure on the attached router, producing a new Link State Packet (LSP) describing the failure, flooding this new LSP and finally updating the Forwarding Information Base (FIB) in all the routers using the failed resources in the network. Sub-second convergence has been made possible, but the sub-50 msec target can only be achieved by the means of a local restoration scheme. Achieving very fast convergence in an IP network will thus require temporary tunnels to quickly reroute traffic around failures, as in MPLS networks [7]. Several solutions to establish such local protections have been proposed in the literature [8], [9], [10], [11], [12]. Unfortunately, in an IP network, using a protection tunnel to locally reroute the traffic around the failed link is not sufficient as transient loops may occur during the update of the FIBs of the other routers in the network.

To understand this problem, let us consider the Inter-

the protection tunnel, so that KC will decapsulate the packet, and forward it to its destination, DN.

This suboptimal routing should not last long, and thus after a while the routers must converge, i.e., adapt to the new shortest paths inside the network, and remove the tunnel. As the link is protected, the reachability of the destinations is still ensured and thus the adaptation to the topological change should be done by avoiding transient loops rather than by urging the updates on each router. The new LSP generated by IP indicates that IP is now only connected to CH and AT. Before the failure, the shortest path from WA to KC, DN, ST and SV was via NY, CH and IP. After the failure, NY will send its packets to KC, DN, ST and SV via WA, AT and HS. During the IGP convergence following the failure of link KC-IP, transient loops may occur between NY and WA depending on the order of the forwarding table updates performed by the routers. If NY updates its FIB before WA, the packets sent by NY to KC via WA will loop on the WA-NY link. To avoid causing a transient loop between WA and NY, WA should update its

[1] A preliminary version of this paper was presented at INFOCOM 2005 [1].

[2] This network is much smaller than large ISP backbones, but it is one of the few networks whose detailed topology is publicly available. We verified that similar transient loops could occur in larger ISP backbones, but the size of those backbones prevented us from using them as an example in this paper. Note that the IGP metrics have been rounded off to facilitate the understanding of the topology. The round off does not influence the routing tables of the network.

Avoiding transient loops

IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 19, NO. 4, AUGUST 2011 1097

Graceful Network State Migrations

Saqib Raza, *Member, IEEE*, Yuanbo Zhu, and Chen-Nee Chuah, *Senior Member, IEEE*

***Abstract*—A significant fraction of network events (such as topology or route changes) and the resulting performance degradation stem from premeditated network management and operational tasks. This paper introduces a general class of *Graceful Network State Migration* (GNSM) problems, where the goal is to discover the optimal sequence of operations that progressively transition the network from its *initial* to a desired *final* state while minimizing the overall performance disruption. We investigate two specific GNSM problems: 1) Link Weight Reassignment Scheduling (LWRS) studies the optimal ordering of link weight updates to migrate from an existing to a new link weight assignment; and 2) Link Maintenance Scheduling (LMS) looks at how to schedule link deactivations and subsequent reactivations for maintenance purposes. LWRS and LMS are both combinatorial optimization problems. We use dynamic programming to find the optimal solutions when the problem size is small, and leverage ant colony optimization to get near-optimal solutions for large problem sizes. Our simulation study reveals that judiciously ordering network operations can achieve significant performance gains. Our GNSM solution framework is generic and applies to similar problems**

I for mission-critical applications and services such as Voice-over-IP, virtual private networks (VPNs), e-commerce applications, and multimedia streaming. Such applications rely upon consistent quality-of-service (QoS) provisioning by Internet service providers (ISPs), with five-nines availability (99.999% uptime) becoming the norm rather than the exception. The end-to-end perceived QoS can potentially be affected due to the dynamic nature of networks. For instance, network topology may change due to transient router/link outages or long-term network engineering. Furthermore, protocol configuration parameters may be altered to migrate from one setting to another. Ideally, QoS guarantees should persist across such dynamic conditions.

Some of these dynamic changes are *inadvertent*, e.g., ones due to faulty interfaces, router crashes, and accidental fiber cuts. However, other changes ensue from deliberate and *premeditated* actions of network operators (e.g., routine maintenance). A failure characterization study of an IP backbone [18] observed that planned maintenance activities account for more than 20% of transient failures. Other studies [8] have also observed the prevalence of such planned maintenance activities. Premeditated network tasks also include network upgrade activities such as adding new routers or overhauling link capacity. Another example of a premeditated network task is migrating an existing OSPF [20] or IS-IS [25][1] link weight assignment to a new assignment that has been optimized based on the most up-to-date traffic matrix estimates.

In the case of premeditated tasks, network operators have the prerogative to decide the sequence of *atomic* operations that comprise such a task. This paper[2] introduces a general

A. Link Weight Reassignment Scheduling

Setting link weights is the primary tool used by network operators to control network load distribution and traffic engineering [9]–[11], [22]. Link weights are optimized based on an estimate of the traffic matrix. They are usually not modified in response to short-term fluctuations in the traffic matrix. However, the estimated traffic matrix may change significantly over a longer period of time, prompting network operators to reoptimize and reset link weights. In such a case, network operators need to migrate from one weight setting to another. The sequence in which the link weights are changed determines the disruption to network traffic during this migration process.

We illustrate this with the help of a toy example. Fig. 1 gives a network with the arc labels representing IGP link weights. Suppose all links have capacity c, and traffic demands between node pairs (a, g) and (b, g) are both $(1/4)c$. The traffic demand between all other node pairs is 0. The link weights depicted in Fig. 1 are optimal for such a traffic matrix given the objective of minimizing the maximum link utilization (MLU). Now suppose that the traffic demand between node pair (b, g) increases to $(3/4)c$. Shortest-path routing using Equal Cost Multi-

Manuscript received June 23, 2009; revised June 16, 2010; accepted November 15, 2010; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Z. M. Mao. Date of publication January 10, 2011; date of current version August 17, 2011.

S. Raza and Y. Zhu are with the Data Center Switching Technology Group, Cisco Systems, San Jose, CA 95134 USA (e-mail: sraza@ucdavis.edu; juzhu@ucdavis.edu).

C.-N. Chuah is with the Department of Electrical and Computer Engineering, University of California, Davis, Davis, CA 95616 USA (e-mail: chuah@ucdavis.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2010.2097604

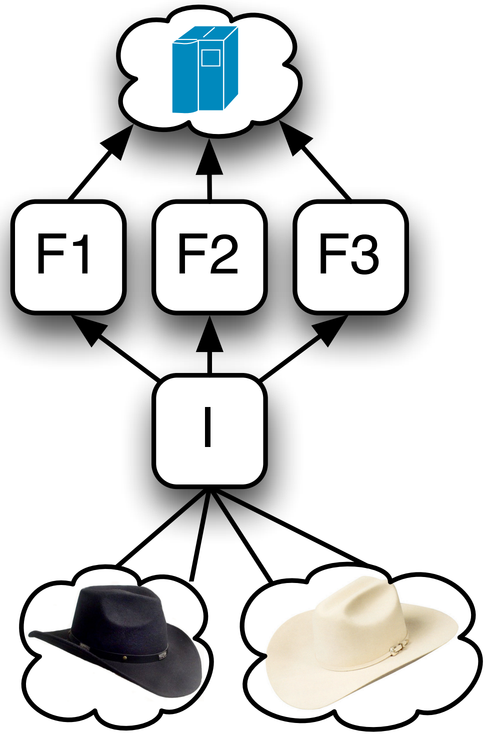
[1] Most common intradomain (IGP) protocols.

[2] This paper is an extended version of our previous work [28].

1063-6692/$26.00 © 2011 IEEE

Graceful state migration

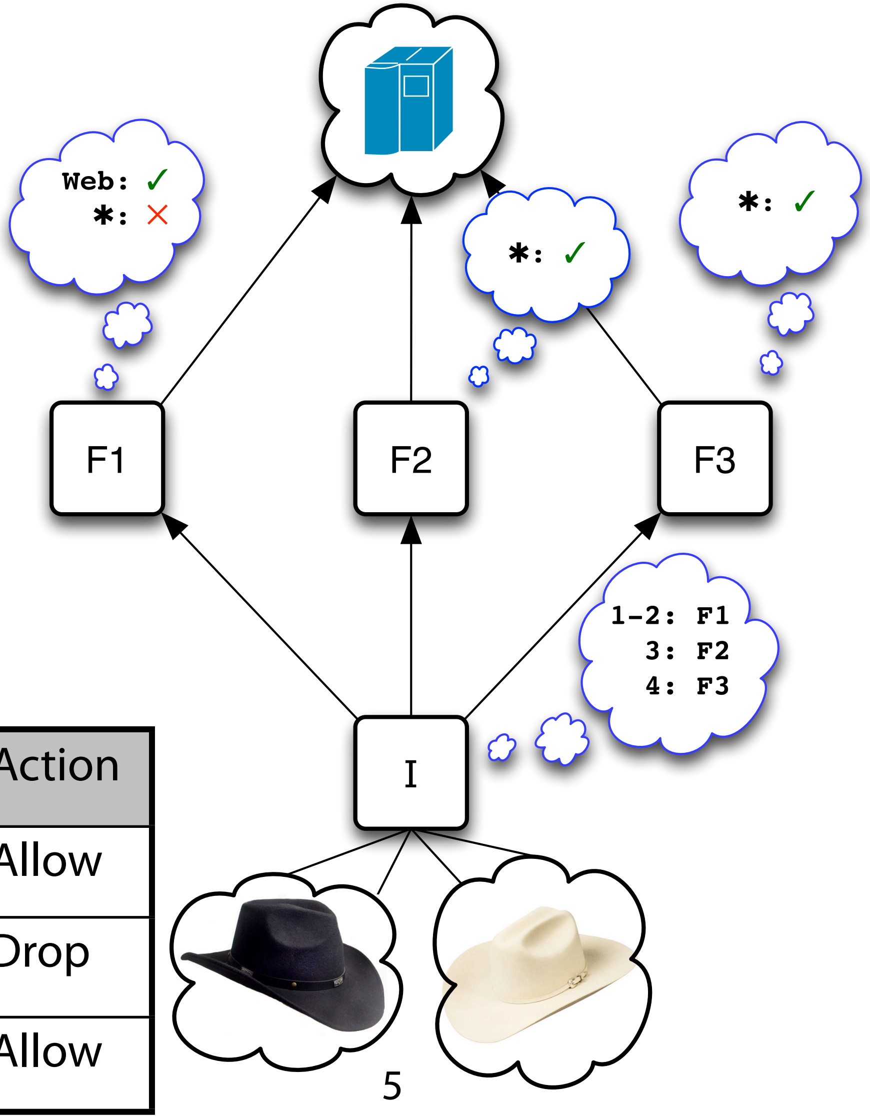
Example



Security Policy

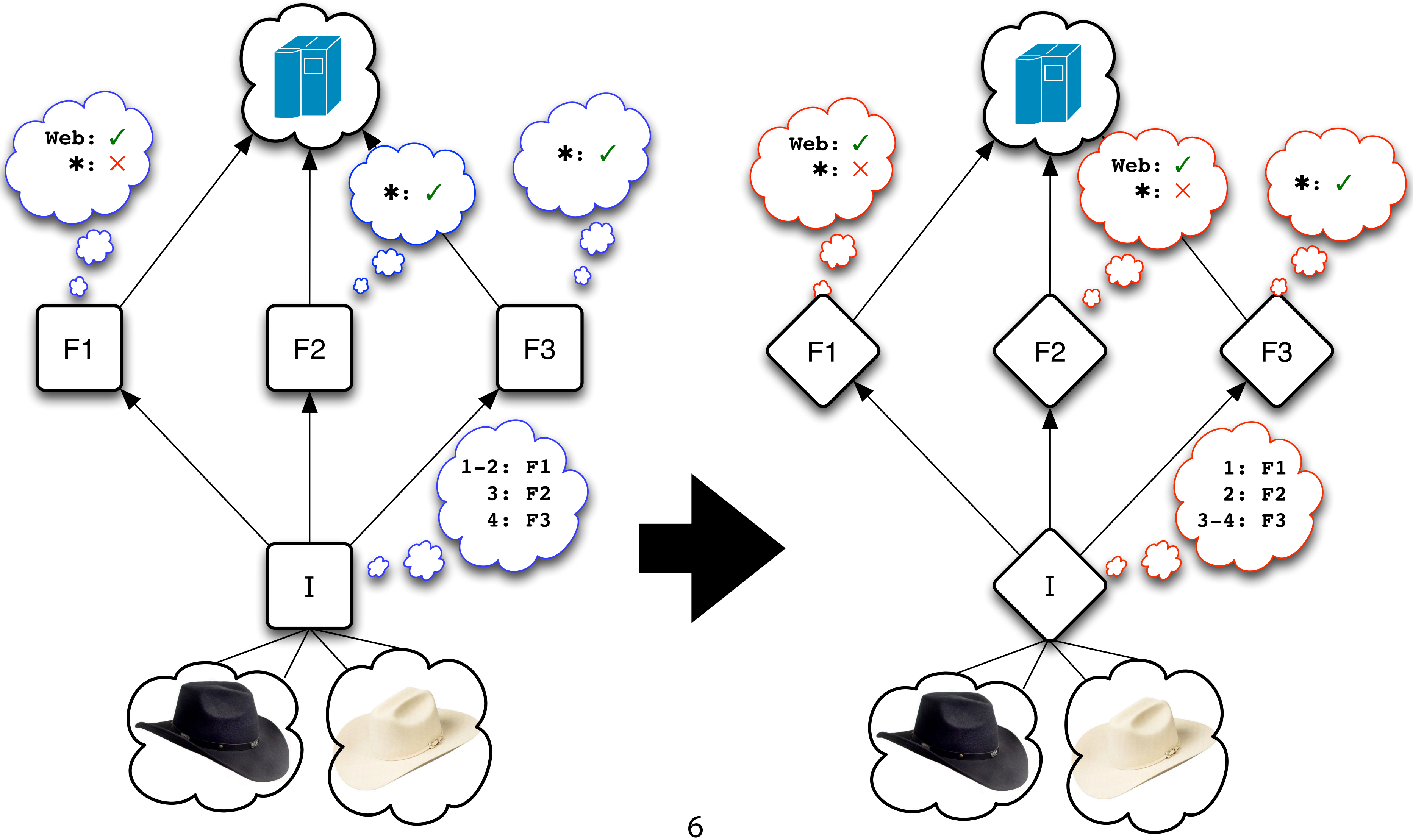
Src	Traffic	Action
	Web	Allow
	Other	Drop
	Any	Allow

Initial Configuration

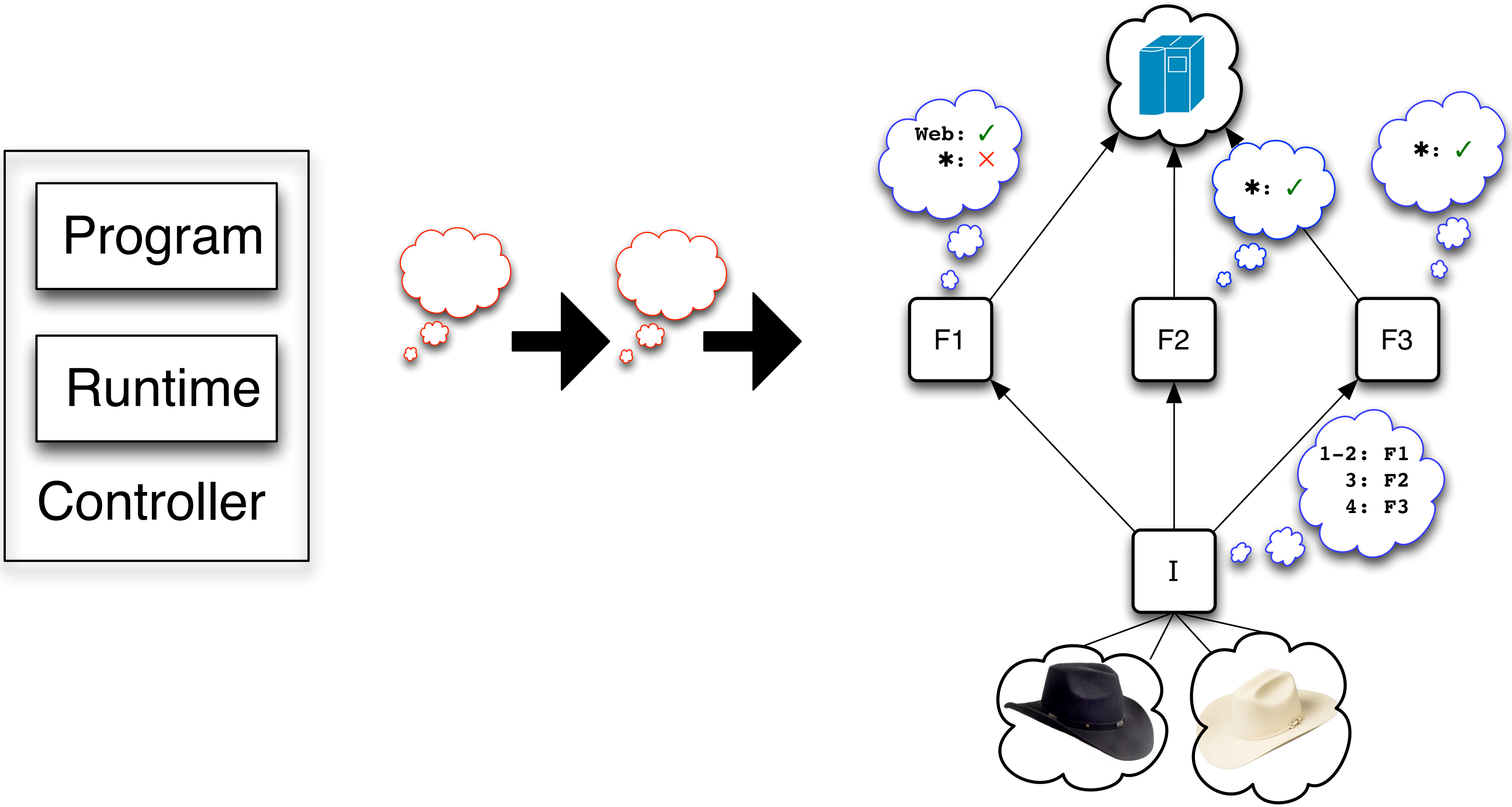


Src	Traffic	Action
	Web	Allow
	Other	Drop
	Any	Allow

Redistribute Configuration



Software Defined Networks (SDN)



Program

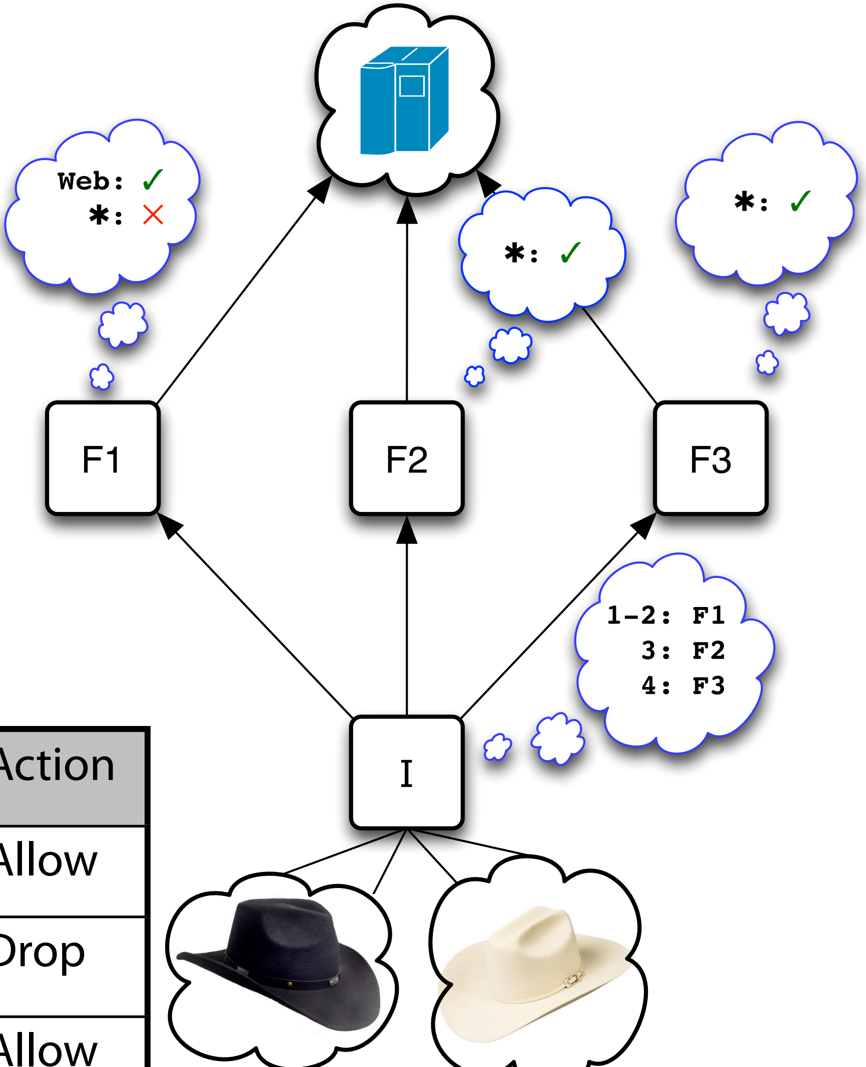
Runtime

Controller

SDN Program

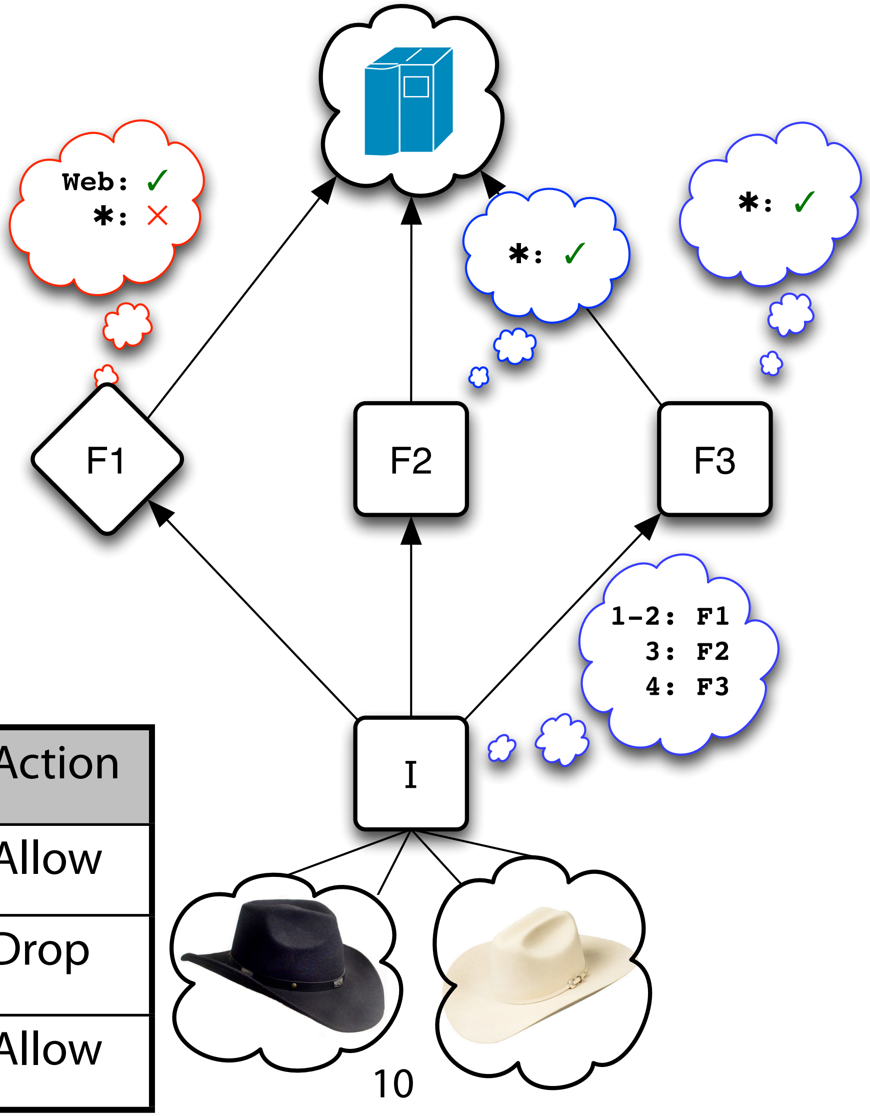
```
void main() {
... monitor ...
Conf = balance_load();
install(F1, Conf[F1]);
install( I, Conf[I]) ;
...
}
```

Initial Configuration



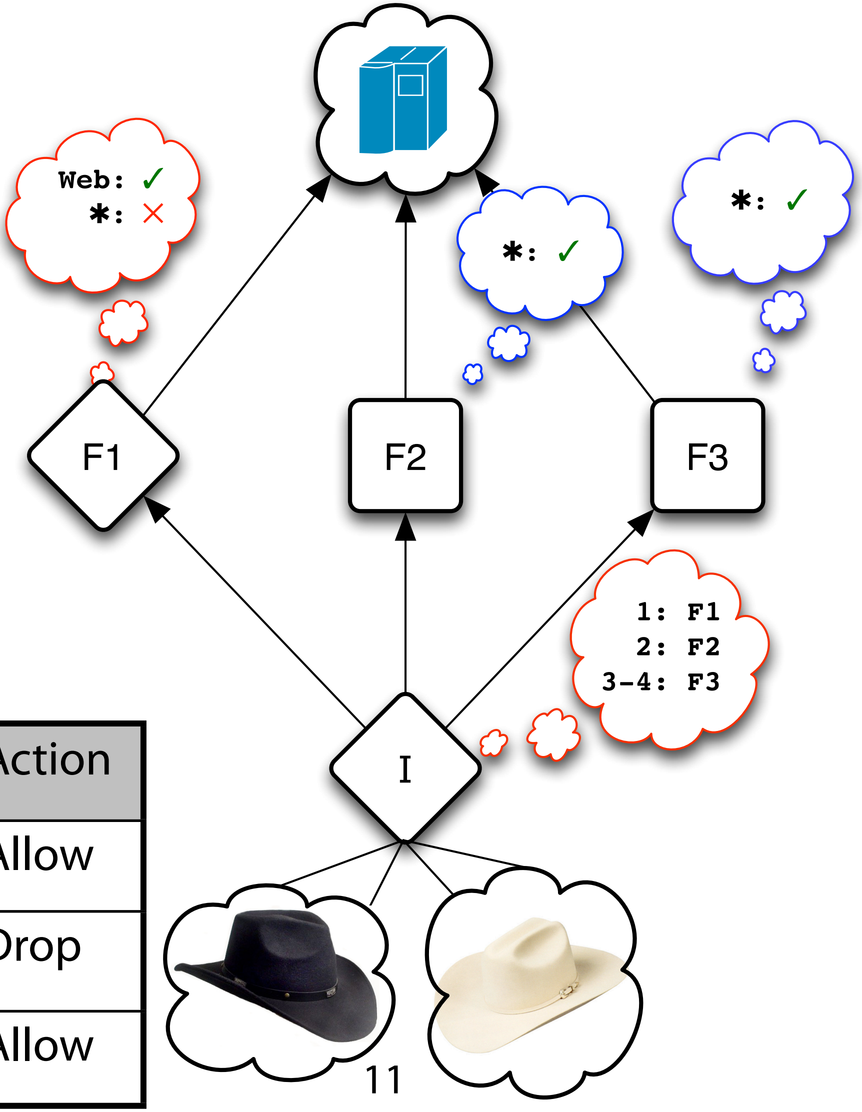
Src	Traffic	Action
	Web	Allow
	Other	Drop
	Any	Allow

Initial Configuration



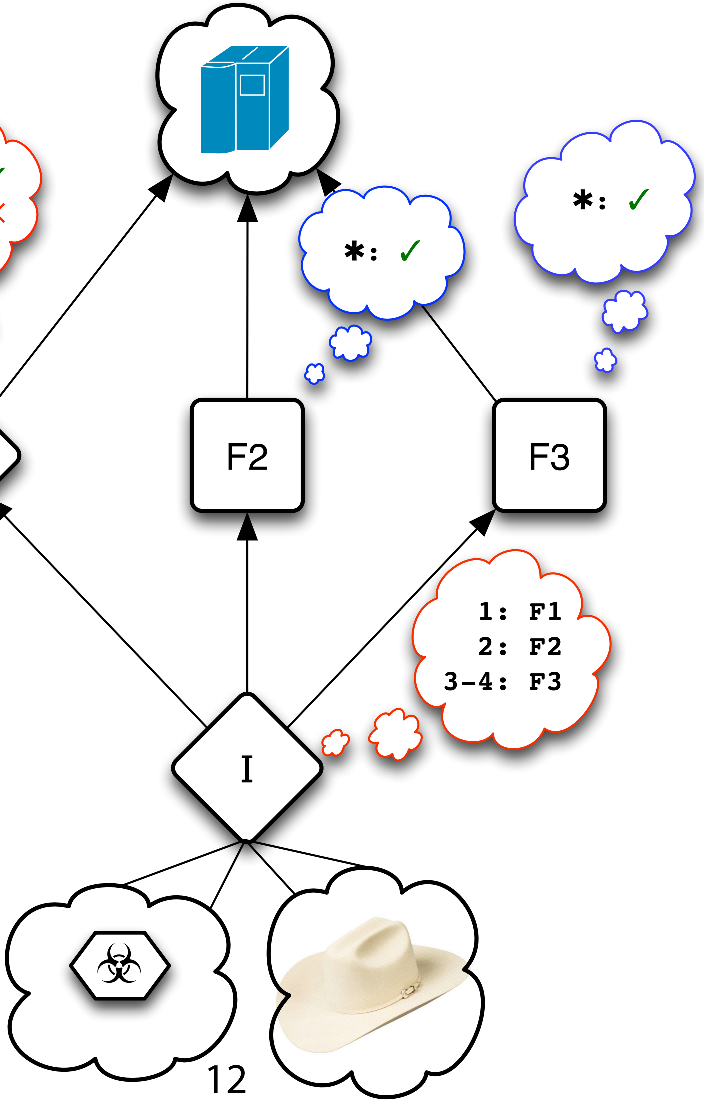
Src	Traffic	Action
	Web	Allow
	Other	Drop
	Any	Allow

Updating Configuration



Src	Traffic	Action
	Web	Allow
	Other	Drop
	Any	Allow

Updating Configuration



Web: ✓
*: ✗

*: ✓

*: ✓

F1

F2

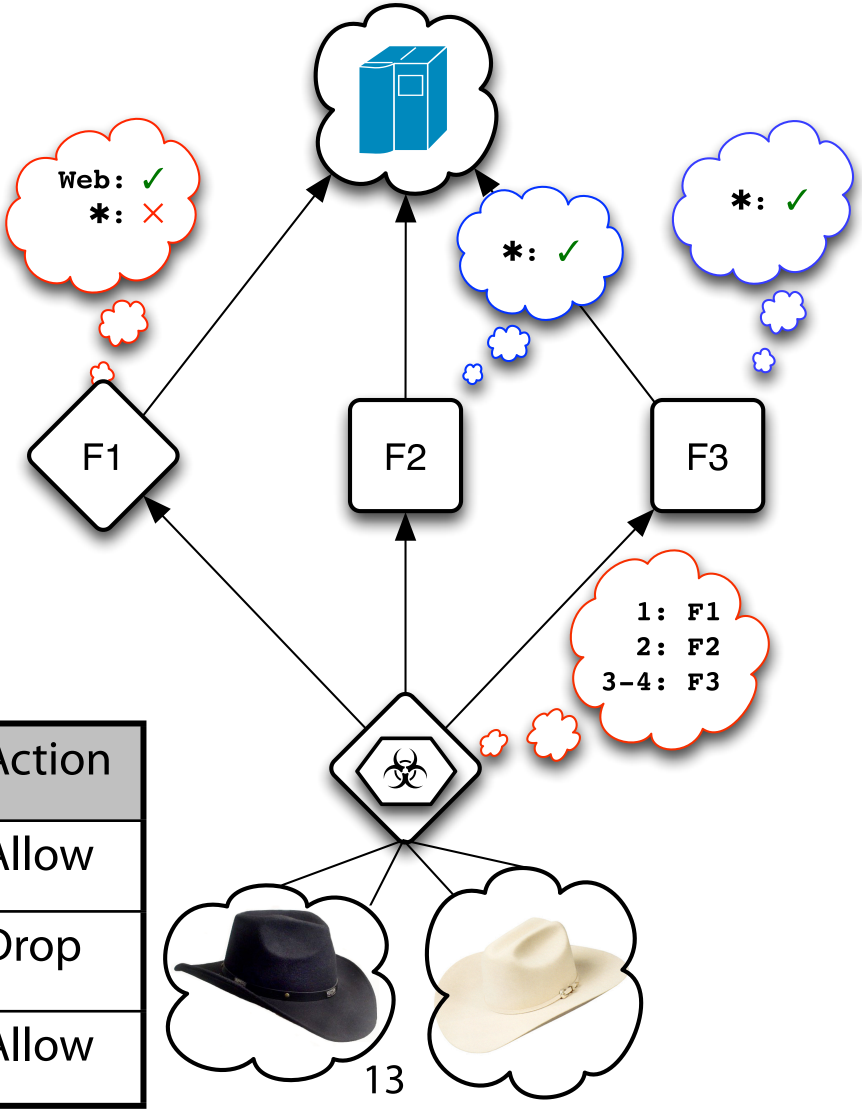
F3

1: F1
2: F2
3-4: F3

I

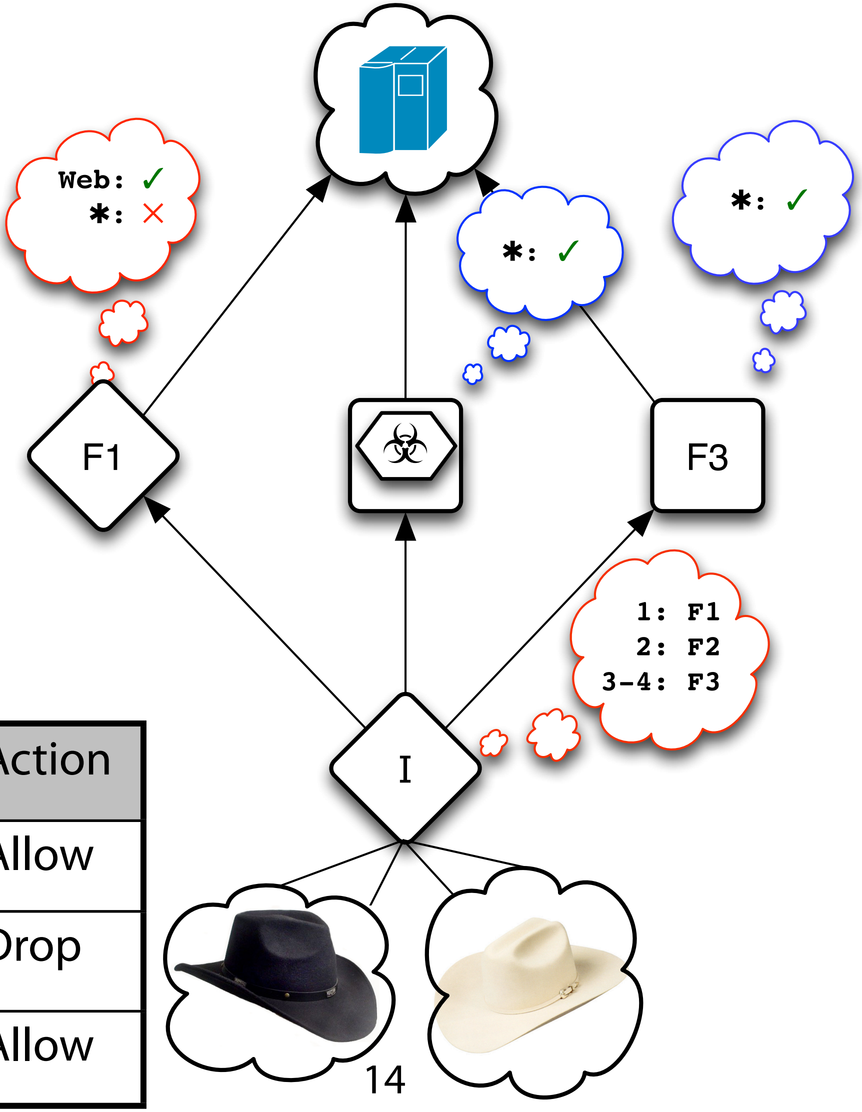
Src	Traffic	Action
	Web	Allow
	Other	Drop
	Any	Allow

Updating Configuration



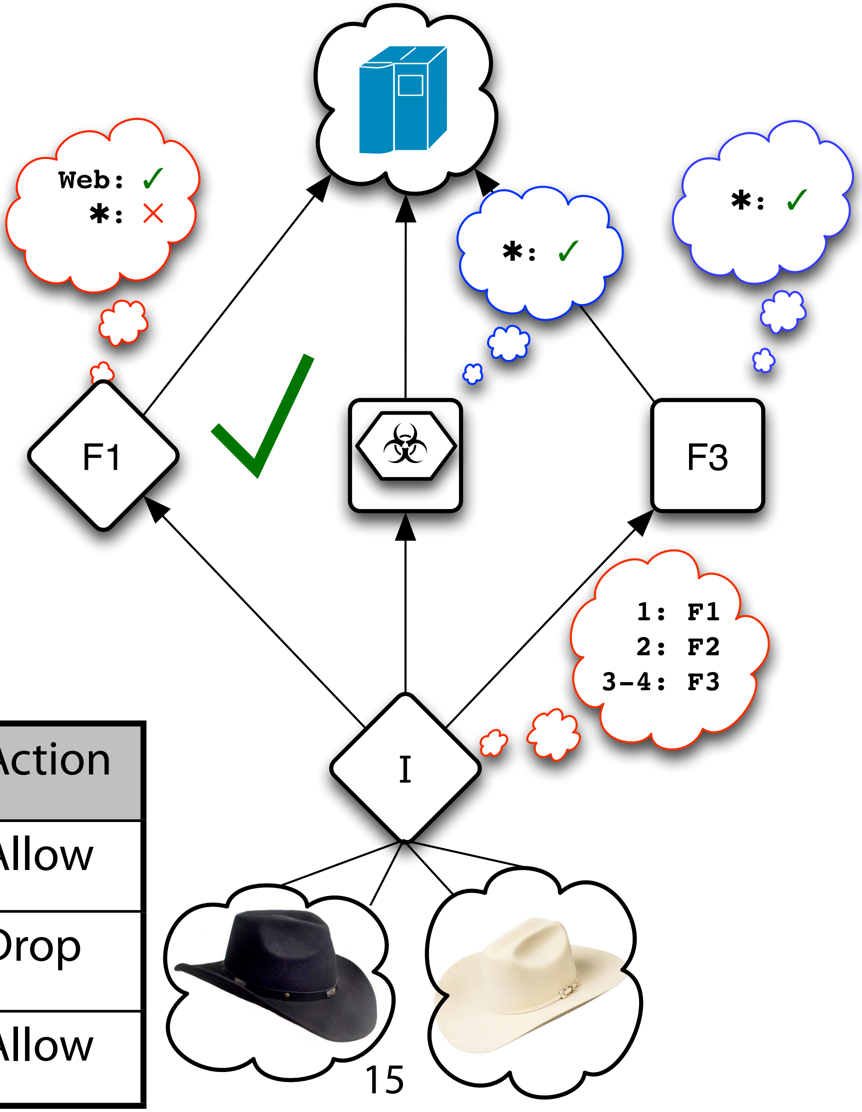
Src	Traffic	Action
	Web	Allow
	Other	Drop
	Any	Allow

Updating Configuration



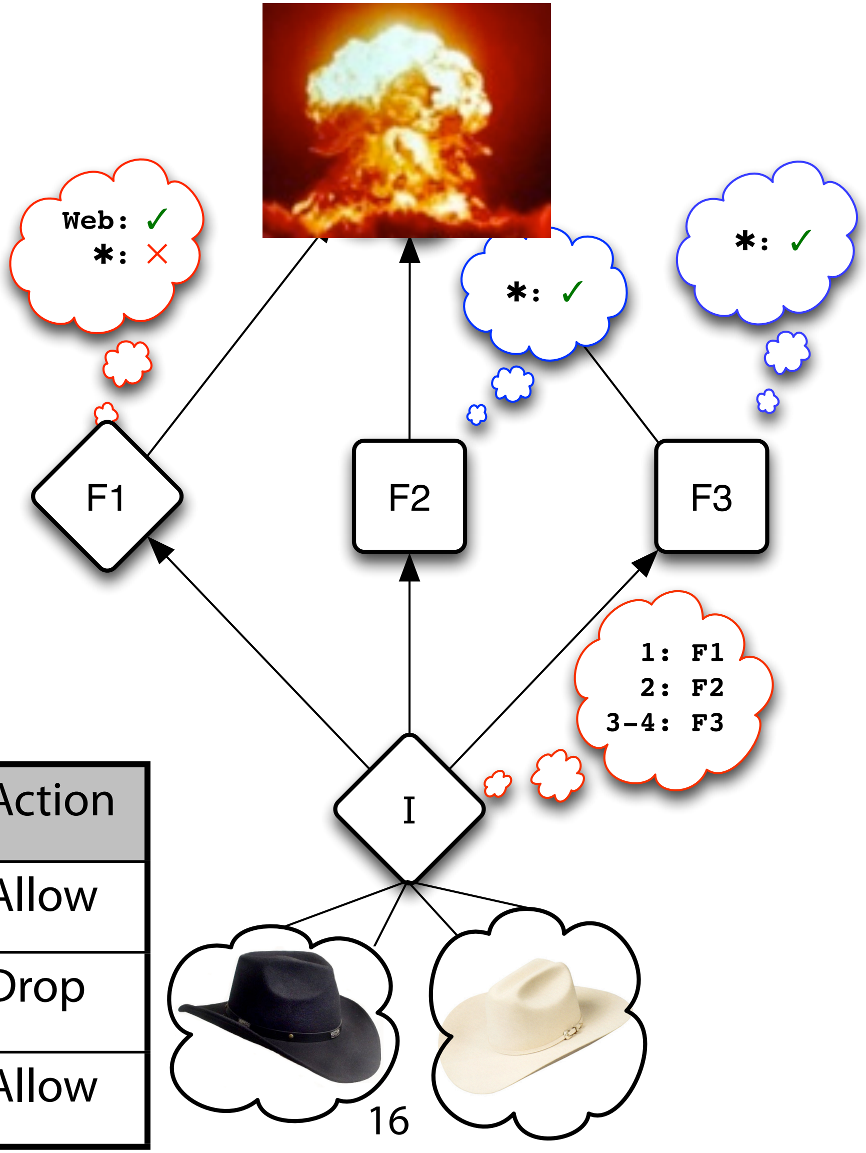
Src	Traffic	Action
	Web	Allow
	Other	Drop
	Any	Allow

Updating Configuration



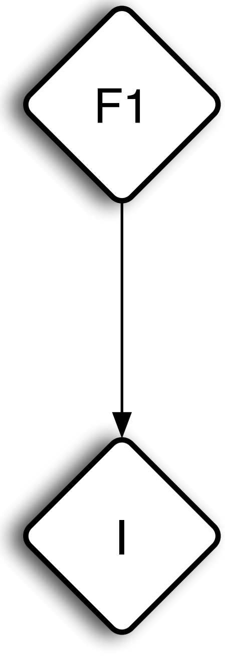
Src	Traffic	Action
	Web	Allow
	Other	Drop
	Any	Allow

Updating Configuration

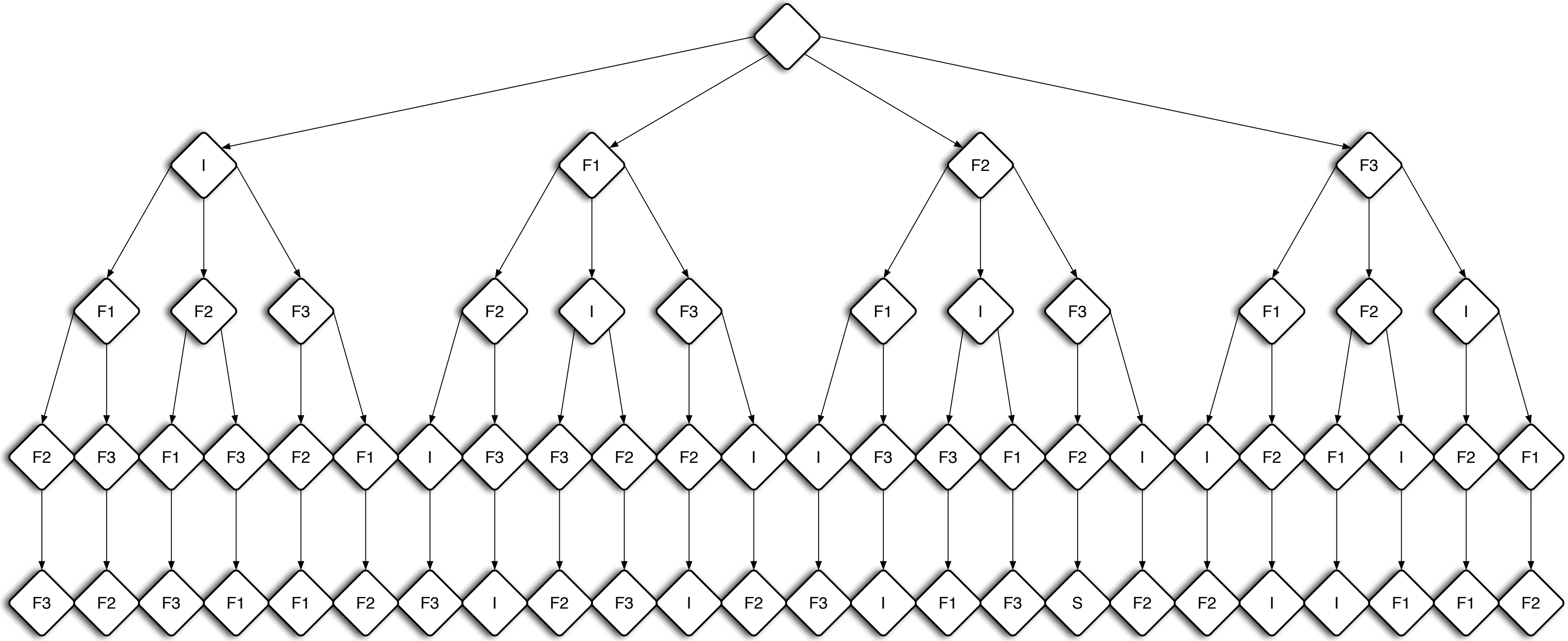


Src	Traffic	Action
	Web	Allow
	Other	Drop
	Any	Allow

Bad Update Order



Bad Update Order

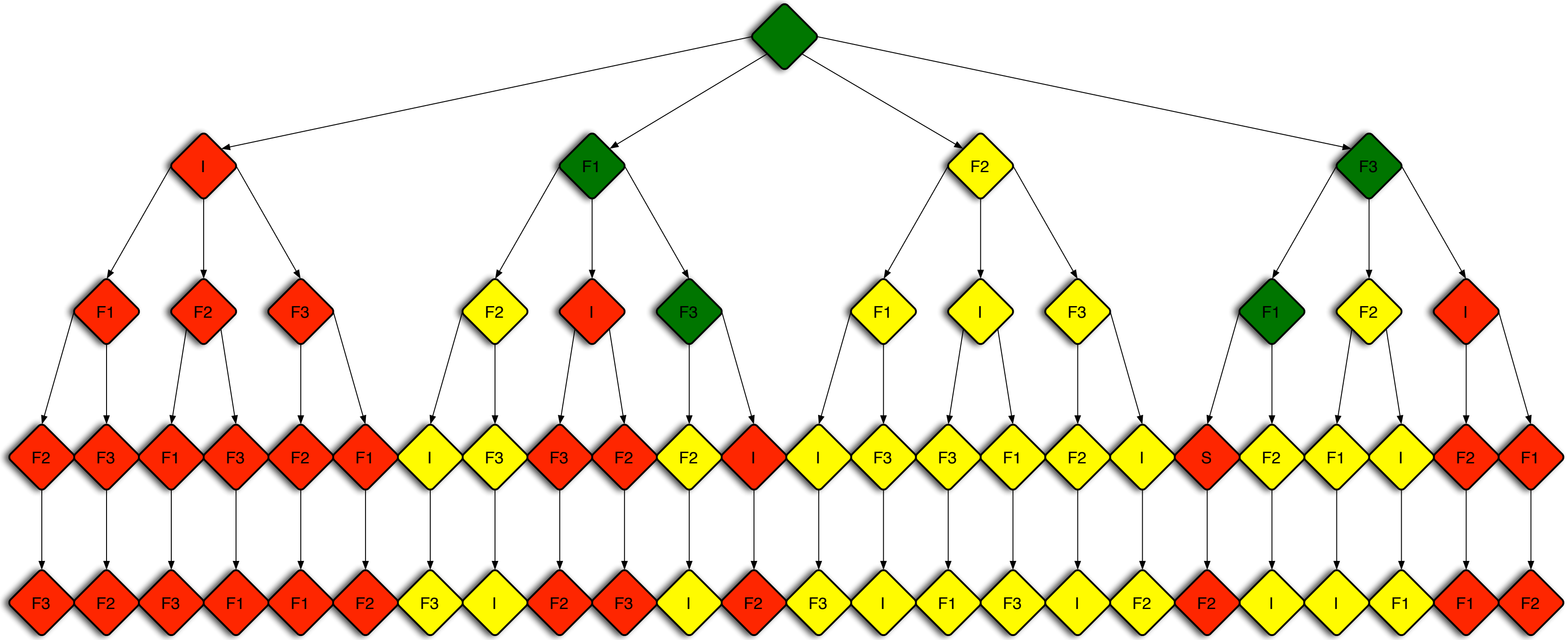

I F1 F2 F3
F1 F2 F3 F2 I F3 F1 I F3 F1 F2 I
F2 F3 F1 F3 F2 F1 I F3 F3 F2 F2 I I F3 F3 F1 F2 I I F2 F1 I F2 F1
F3 F2 F3 F1 F1 F2 F3 I F2 F3 I F2 F3 I F1 F3 S F2 F2 I I F1 F1 F2

Bad Update Order

🟩 Safe

🟨 Broken Connectivity

🟥 Broken Security



Towards a Solution

Towards a Solution

Updating individual switches doesn’t work!

Towards a Solution

Updating individual switches doesn’t work!

What’s the solution?

- Don’t implement updates rule-by-rule and switch-by-switch!

Towards a Solution

Updating individual switches doesn’t work!

What’s the solution?

- Don’t implement updates rule-by-rule and switch-by-switch!
- Leverage the run-time system to handle tedious, low-level details

SDN Program

```
void main() {
... monitor ...
Conf = balance_load();
install(Conf);
}
```

Per-packet Consistency

An update from configuration A *to configuration* B *is* **per-packet consistent** *if each packet is routed according to either configuration* A *or* B.

Path Properties

A ***path property*** ϕ *specifies the legal paths that a packet can take through a network* **N**.

Formally: $\phi \subseteq$ **Packet** x **Paths(N)**.

- Loop-freedom
- "Block SSH traffic from 10/8"
- "All Web traffic waypoints through switch 5"

SDN Program

```
void main() {
... monitor ...
Conf = balance_load();
install(perpacket, Conf);
}
```

Beyond Path Properties

Not path properties:

- In-order delivery
- Flow affinity

An update from configuration A *to configuration* B *is* **per-flow consistent** *if each packet* ***in the same flow*** *is routed according to either configuration* A *or* B.

See paper for details

2-Phase Implementation

1. Instrument new configuration with version
2. Install instrumented configuration, leaving all old ingress rules active
3. Activate new ingress rules
4. Wait for old version packets to leave
5. Uninstall old configuration

Future Work

Implementation

- Naive prototype running
- Exploring performance optimizations

Unplanned Change

- Highly responsive
- Weaker consistency

Formal Verification

- Specification language for path properties
- Configuration verifier

Ongoing Work

frenetic >>

Ongoing Work

frenetic >>

- **This paper**

  Network write abstraction

Ongoing Work

frenetic >>

- **This paper**

  Network write abstraction

- **PRESTO '10, ICFP '11**

  Network read abstraction

Ongoing Work

frenetic >>

- **This paper**
  Network write abstraction

- **PRESTO ’10, ICFP ’11**
  Network read abstraction

- **POPL ’12**

Ongoing Work

frenetic >>

- **This paper**
  Network write abstraction

- **PRESTO ’10, ICFP ’11**
  Network read abstraction

- **POPL ’12**
  Rich policy abstraction

Questions?

Thank You

frenetic >>

http://frenetic-lang.org

Database Analogy

Network	Database
Fully routed packet	Read Transaction
Single hop routed packet	Read
Network update	Write Transaction
Single switch update	Write
Per-Packet Consistency	Snapshot Isolation